**Product Requirements Document (PRD): Retail AI Advisor (Hackathon MVP)**

Document Version: 1.0

Date: May 31, 2025

Prepared For: Hackathon Judges & Internal Team

Prepared By: AI Assistant (on behalf of the Solo Developer)

**Introduction / Executive Summary**

The Retail AI Advisor is an innovative Minimum Viable Product (MVP) developed during a focused 32-hour hackathon. Its core purpose is to empower small to medium-sized (SME) e-commerce retailers, particularly those on Shopify, with actionable, data-driven insights to optimize product pricing, manage inventory, and capitalize on market trends. By integrating diverse data sources – Shopify sales and product data, competitor pricing, Google Trends, and accounting costs – the system identifies "underpriced," "overstocked," and "hot" products. A unique aspect of this MVP is its use of generative AI to transform these complex data insights into concise, personalized video summaries, delivered directly to the retailer.

This document details the product's purpose, key features, comprehensive data integration strategy, chosen technology stack, an ambitious hackathon timeline, success criteria, explicit assumptions and constraints, and a vision for future development. Notably, this entire project is being developed by a single developer leveraging advanced AI tools for all coding, demonstrating a cutting-edge approach to rapid prototyping and feature delivery.

**Section 1: Purpose & Target Audience**

This section articulates the fundamental reason for the product's existence, the problem it solves, and the specific group of users it aims to serve.

**1.1 Problem Statement**

Small to medium-sized (SME) e-commerce retailers often operate with limited resources and lack access to sophisticated data analytics and market intelligence tools available to larger enterprises. This results in:

* **Suboptimal Pricing:** Products may be underpriced (leaving money on the table) or overpriced (losing sales) due to a lack of competitor awareness and real-time market insights.
* **Inefficient Inventory Management:** Overstocking leads to tied-up capital and potential write-offs, while understocking results in lost sales and customer dissatisfaction.
* **Missed Market Opportunities:** Retailers struggle to identify trending products or shifts in consumer interest, leading to a reactive rather than proactive approach to merchandising.
* **Information Overload:** Even with some data, converting raw numbers into clear, actionable insights is time-consuming and requires analytical expertise.

**1.2 Solution Overview**

The Retail AI Advisor aims to solve these problems by providing an intuitive, automated system that:

1. Aggregates key e-commerce data (sales, inventory, product costs, competitor prices, market trends).
2. Applies intelligent algorithms to identify critical business insights.
3. Delivers these insights in an easily digestible, actionable format, including unique, personalized video summaries.

**1.3 Product Goals (for Hackathon MVP)**

* **Validate the concept** of AI-driven, personalized video insights for retailers.
* **Prove technical feasibility** of integrating diverse e-commerce data sources within a short timeframe.
* **Demonstrate core value** by generating actionable pricing and inventory recommendations.
* **Showcase rapid development** through efficient use of AI coding tools.

**1.4 Target Audience**

The primary target audience for the Retail AI Advisor MVP is:

* **Small to Medium-sized E-commerce Retailers:** Specifically those using **Shopify** as their primary sales channel.
* **Non-Technical Business Owners:** Retailers who understand their products but may lack the time, skills, or resources for deep data analysis.
* **Growth-Oriented Businesses:** Retailers actively seeking data-driven strategies to increase revenue, optimize profits, and improve operational efficiency.
* **Early Adopters of AI:** Businesses curious about leveraging AI to gain a competitive edge.

**Exclusions for MVP:** Large enterprises with complex ERP systems, retailers not on Shopify (unless using the CSV fallback for demo purposes), or those primarily focused on brick-and-mortar sales.

**Section 2: Key Features (Minimum Viable Product - MVP)**

This section outlines the essential functionalities that will be developed and delivered within the hackathon timeframe to achieve the product's core goals.

**2.1 Core Functionality**

* **Shopify Store Connection & Data Sync:**
  + **OAuth 2.0 Integration:** Secure connection to a Shopify store via OAuth, allowing the application to read product, inventory, and order data.
  + **Initial & Incremental Data Sync:** Automated daily batch synchronization of historical sales data (last 12 months) and all active product/inventory details from Shopify.
  + **CSV Fallback:** Ability to manually upload CSV files for Product/Inventory and Sales Data for non-Shopify retailers or initial data load, with strict validation.
* **Competitor Pricing Insights:**
  + **Configurable Scraping:** Backend process to scrape competitor product prices from specified URLs using ZenRows, identified by product SKUs.
  + **Min/Max Price Aggregation:** Calculation and storage of the minimum and maximum competitor prices for each product SKU.
* **Market Trend Insights:**
  + **Google Trends Integration:** Automated fetching of Google Trends data for product titles (as keywords) via pytrends.
  + **Mock Social Mention Score:** For MVP, a mocked "social score" (random value 20-80) to simulate product social media engagement.
  + **Unified Trend Score:** Combination of Google Trends data and mock social score into a single final\_score and an associated label (e.g., "Hot," "Rising," "Steady," "Declining").
* **Cost of Goods Sold (COGS) Integration:**
  + **API Deck Integration:** Connect to popular accounting software (initially QuickBooks Online / Xero via API Deck) to retrieve product cost\_price based on SKU matching.
  + **Cost Price Hierarchy:** Prioritize cost\_price from accounting software; fall back to Shopify's variant.cost, and finally default to 0.00 if no cost is found.
* **Pricing Engine:**
  + **Rule-Based Recommendations:** Calculate a recommended\_price for each product SKU based on:
    - **Cost-Plus Pricing:** Ensuring a minimum configurable margin (e.g., 30% above cost\_price).
    - **Competitor Matching:** Adjusting price to be competitive (e.g., matching min\_price from competitors, or 5% above min\_price).
    - **Trend Adjustment:** Slightly higher recommendation for "Hot" or "Rising" products, lower for "Declining" products.
  + **Justification:** Generate a textual pricing\_reason for each recommendation (e.g., "Underpriced compared to competitors," "High trend score allows for price increase").
* **Trend Engine:**
  + Classify products into distinct trend categories (Hot, Rising, Steady, Declining) based on their final\_score from market trend analysis.
* **Generative Media - AI Advisor Video:**
  + **Insight Aggregation:** Backend logic to identify and select the **top 3 most impactful insights** from the processed data (e.g., top 1-2 "underpriced" products, top 1-2 "overstocked" products, top 1-2 "hot" products).
  + **Script Generation:** Use Azure OpenAI's GPT-4 Turbo model to generate a concise, engaging script (45-60 seconds) summarizing these top 3 insights, tailored for a business owner.
  + **Video Generation:**
    - **Text-to-Speech:** Utilize ElevenLabs API to convert the script into natural-sounding voiceover audio.
    - **Video Composition:** Use VEED.io API to combine the voiceover with a basic, pre-defined visual template (e.g., stock footage, simple text overlays, product images for each insight) to create a short MP4 video.
  + **Video Delivery:** Store the generated video URL and transcript in Supabase, accessible from the frontend dashboard.
* **Frontend Dashboard:**
  + **Secure Login:** User authentication via Supabase Auth (Email/Password).
  + **Product Overview:** Display a clear, interactive table of all synced products showing: SKU, Title, Current Price, Inventory Level, Cost Price, Competitor Price Range, Recommended Price, and Trend Label.
  + **Filtering & Searching:** Basic search and filter capabilities for products.
  + **Advisor Video Playback:** Embed a video player to play the latest generated AI Advisor video directly within the dashboard.

**2.2 Non-Goals for MVP**

* **Real-time synchronization:** All data updates are batch-based, not webhook-driven.
* **Multi-currency support:** All pricing assumed to be in GBP for MVP.
* **Complex financial reporting:** Only COGS for SKU matching is in scope.
* **User-configurable advanced pricing models:** Rule-based logic is hardcoded.
* **Automated price pushing back to Shopify:** Recommendations are advisory only.
* **Customer segmentation or personalized marketing automation.**
* **Refund/Discount reconciliation** for sales data.
* **Advanced analytics dashboards** with drill-downs or custom report generation.
* **Mobile-native application** (focus on responsive web for demo).
* **Deep social listening integration** (mocked for MVP).

**Section 3: Data Source Specification & Integration Strategy**

This section outlines the detailed specification for all external data sources integrated into the system, defining their exact requirements, integration methods, ingestion frequencies, and how the data is ultimately stored and utilized.

**3.1 Product & Sales Data Collection**

This subsection details the primary and fallback mechanisms for ingesting core e-commerce data (product catalog, inventory, and sales records).

* **3.1.1 Primary Source: Shopify Integration**
  + **Authentication & Setup:**
    - **OAuth Flow:** Standard 3-legged OAuth process (User Authorizes -> Shopify Redirects to Backend with code -> Backend exchanges code for access\_token -> Backend stores access\_token securely in Supabase stores table).
    - **Minimum Required Shopify API Scopes:** read\_products, read\_inventory, read\_orders.
    - **Secure Storage of API Key/Access Token:** access\_token stored in Supabase stores table with RLS; client\_id and client\_secret as environment variables on FastAPI server (Azure Key Vault in production).
    - **Shopify API Version:** 2024-07.
  + **Data Extraction - Product Catalog:**
    - **Shopify API Endpoint:** GET /admin/api/2024-07/products.json?status=active&limit=250 (paginated).
    - **Exact Fields Mapped to Supabase** products **table:**
      * product.id -> shopify\_product\_id (BIGINT, NOT NULL, UNIQUE)
      * product.title -> product\_title (TEXT, NOT NULL)
      * variant.id -> sku\_id (BIGINT, NOT NULL, PRIMARY KEY) - *Our internal SKU ID*
      * variant.sku -> sku\_code (TEXT, NOT NULL, UNIQUE) - *Human-readable SKU*
      * variant.title -> variant\_title (TEXT)
      * variant.price -> current\_price (NUMERIC(10,2), NOT NULL)
      * variant.inventory\_quantity -> inventory\_level (INTEGER, NOT NULL)
      * variant.cost -> cost\_price (NUMERIC(10,2), DEFAULT 0.00)
      * product.image.src (first image) -> image\_url (TEXT)
      * product.status -> status (TEXT, e.g., 'active', 'archived')
      * product.created\_at -> created\_at (TIMESTAMP WITH TIME ZONE)
      * product.updated\_at -> updated\_at (TIMESTAMP WITH TIME ZONE)
      * shop\_id (Foreign Key to stores table) (BIGINT, NOT NULL)
    - **Fallback for** variant.cost**:** Defaults to 0.00 if null, later reconciled by API Deck.
    - **Handling Deleted Products/Variants:** Soft-deleted by setting status = 'archived' or status = 'inactive'.
  + **Data Extraction - Inventory Levels:**
    - Primarily from products.json's variant.inventory\_quantity.
    - **Rate Limits:** Implement backoff strategy based on Shopify headers.
    - **Expected Frequency:** Initial sync on store connection, daily incremental updates via cron job.
  + **Data Extraction - Sales Data:**
    - **Shopify API Endpoint:** GET /admin/api/2024-07/orders.json?status=any&limit=250&created\_at\_min={timestamp}.
    - **Exact Fields Mapped to Supabase** sales **table:**
      * order.id -> shopify\_order\_id (BIGINT, NOT NULL)
      * line\_item.id -> shopify\_line\_item\_id (BIGINT, NOT NULL)
      * line\_item.sku -> sku\_code (TEXT, NOT NULL, FK to products.sku\_code)
      * line\_item.quantity -> quantity\_sold (INTEGER, NOT NULL)
      * line\_item.price -> sold\_price (NUMERIC(10,2), NOT NULL)
      * order.created\_at -> sold\_at (TIMESTAMP WITH TIME ZONE, NOT NULL)
      * shop\_id (BIGINT, NOT NULL, FK to stores table)
      * id (BIGINT, PRIMARY KEY, auto-incrementing)
    - **Idempotency & Duplicate Prevention:** UNIQUE (shopify\_order\_id, shopify\_line\_item\_id) on sales table with ON CONFLICT DO NOTHING.
    - **Historical vs. Incremental Sync:** Initial sync from 2024-05-30T00:00:00Z or store creation. Incremental syncs from NOW() - INTERVAL '48 hours'. Refunds out of scope for MVP.
  + **Supabase Schema Impact (SQL):** (See full SQL in previous detailed response for brevity here).
  + **Synchronization Endpoint (**POST /api/shopify-sync**):**
    - **Payload:** {"shop\_id": "...", "access\_token": "..."}.
    - **HTTP Status Codes:** 200 OK (success/warnings), 400 Bad Request, 401 Unauthorized, 500 Internal Server Error.
    - **Expected Frequency:** Daily cron (02:00 AM BST) and manual "Sync Now" button.
* **3.1.2 CSV Fallback (Non-Shopify Retailers / Initial Data Load)**
  + **"Products & Inventory" CSV Specification:**
    - **Header:** SKU,Title,Variant Title,Current Price,Cost Price,Inventory Level,Image URL
    - **Validation:** Strict (e.g., SKU TEXT NOT NULL, Current Price NUMERIC(10,2) NOT NULL >0, Inventory Level INTEGER NOT NULL >=0, URLs valid if present).
  + **"Orders & Sales" CSV Specification:**
    - **Header:** Order ID,SKU,Quantity Sold,Sold Price,Sold At
    - **Validation:** Strict (e.g., Order ID TEXT NOT NULL, SKU must exist in products, Quantity Sold INTEGER NOT NULL >0, Sold Price NUMERIC(10,2) NOT NULL >=0, Sold At ISO 8601 TIMESTAMPTZ).
  + **CSV Upload Flow & Backend Handling:**
    - **Max File Size:** 10MB (Products), 20MB (Orders).
    - **Max Row Count:** 10,000 (Products), 100,000 (Orders).
    - **Encoding:** UTF-8.
    - **Validation Logic:** Backend uses pandas and Pydantic for header and row-level validation. Detailed error messages for invalid rows/format.
    - **Ingestion:** Batched UPSERT operations to Supabase (products) and INSERT (sales).

**3.2 Competitor Pricing via ZenRows**

* **3.2.1 Tool: ZenRows API Specification**
  + **Endpoint:** https://api.zenrows.com/v1/get
  + **Parameters:** url, apikey, js\_render=true, premium\_proxy=true, device=desktop.
  + **API Key Storage:** Azure Key Vault.
* **3.2.2 Competitor URL Configuration (**config/competitor\_urls.json**)**
  + **JSON Schema:** { "SKU123": [{ "competitor\_name": "...", "url": "...", "css\_selectors": ["..."] }] }
  + **Maintenance:** Developer-managed JSON file.
* **3.2.3 Scraping Logic (**scrape\_competitors.py**)**
  + **HTTP/Parsing:** requests, BeautifulSoup4.
  + **CSS Selectors:** Attempted in order.
  + **Currency Cleaning:** replace('£', '').replace('$', '').replace('€', '').replace(',', '').strip(). All GBP for MVP.
  + **Error Handling:** timeout=15s, 3 retries with exponential backoff.
  + **Derivation:** min\_price and max\_price derived from successful scrapes. Null if no prices found. Max 5 URLs per SKU.
* **3.2.4 Output & Ingestion Endpoint (**POST /api/competitor-prices**)**
  + **Payload:** [{"sku\_code": "...", "min\_price": ..., "max\_price": ..., "scraped\_at": "..."}]
  + **Supabase ORM/SQL:** competitor\_prices table. ON CONFLICT (shop\_id, sku\_code) DO UPDATE SET ... for latest entry.
  + **Frequency:** Daily cron job after product data sync.

**3.3 Market Trends (Google Trends + Social Mentions)**

* **3.3.1 Google Trends via pytrends**
  + pytrends **Version:** 4.9.2.
  + **Parameters:** hl="en-GB", tz=0 (UTC), timeframe='now 7-d'.
  + **Keyword:** Product titles.
  + **Batching:** Max 5 keywords per call, random sample if more products.
  + trend\_index**:** Average of daily interest scores over last 7 days.
  + **Error Handling:** try-except, 3 retries with exponential backoff. google\_trend\_index defaults to 0 on failure.
* **3.3.2 Social Mentions (Mocked for MVP)**
  + **Mocked Data:** Random integer 20-80 for social\_score per SKU.
  + **Post-MVP:** Sieve.tool integration planned.
* **3.3.3 Output & Ingestion Endpoint (**POST /api/trends**)**
  + **Payload:** [{"sku\_code": "...", "google\_trend\_index": ..., "social\_score": ..., "final\_score": ..., "label": "...", "computed\_at": "..."}]
  + **Supabase ORM/SQL:** trend\_insights table. ON CONFLICT (shop\_id, sku\_code) DO UPDATE SET ... for latest entry.
  + **Frequency:** Daily cron job after product data sync.

**3.4 Accounting Integration (API Deck)**

* **3.4.1 Tool: API Deck Specification**
  + **Endpoints:** api.apideck.com/v1/accounting/companies/{company\_id}/items (for QuickBooks, Xero).
  + **Authentication:** OAuth 2.0. access\_token stored in stores table.
* **3.4.2 Data Extraction - COGS:**
  + **Fields:** item.product\_id (for SKU match), item.purchase\_details.unit\_cost.
  + **Matching:** Exact match item.product\_id to products.sku\_code.
  + **Priority:** API Deck cost > Shopify variant.cost > 0.00.
* **3.4.3 Output & Ingestion Endpoint (**POST /api/accounting-costs**)**
  + **Payload:** [{"sku\_code": "...", "cost\_price": ..., "source": "...", "synced\_at": "..."}]
  + **Supabase ORM/SQL:** UPDATE products SET cost\_price = :cost\_price WHERE sku\_code = :sku\_code AND shop\_id = :shop\_id; (directly updates products table).
  + **Frequency:** Daily cron job after Shopify sync, before pricing calculations.

**3.5 Unified Data Flow & Orchestration (Shopify-First)**

* **3.5.1 Initial Setup / Onboarding Flow:** User connects Shopify (triggers full historical sync) -> (Optional) User connects Accounting (triggers cost fetch) -> Initial run of Competitor, Trends, Pricing, Trend Engines, and Generative Media.
* **3.5.2 Scheduled Data Ingestion & Processing:** Daily Azure Function/FastAPI background task (02:00 AM BST) orchestrated sequence:
  1. Shopify Data Ingestion (POST /api/shopify-sync)
  2. API Deck Accounting Cost Update (POST /api/accounting-costs)
  3. Competitor Pricing Scraping (scrape\_competitors.py to POST /api/competitor-prices)
  4. Google Trends & Social Trend Calculation (fetch\_trends.py to POST /api/trends)
  5. Pricing Engine Calculation (Backend function)
  6. Trend Engine Calculation (Backend function)
  7. Generative Media Insight Aggregation (Backend, triggers video pipeline)
  8. **Failure Handling:** Log warnings, subsequent stages attempt to run with available data. Critical failures halt pipeline.
  9. **Max Duration:** Targeted <30 minutes for entire daily pipeline.
* **3.5.3 Data Integrity & Reconciliation:** Consistency via UPSERTs and FK constraints. Mismatched SKUs in sales cause warnings. cost\_price prioritized by API Deck.
* **3.5.4 Supabase Schema (Consolidated View):** (See full SQL and ER Diagram in previous detailed response for brevity here).

**Section 4: Technology Stack**

This section defines the precise technical components, programming environments, and deployment strategies chosen for the project, emphasizing the role of AI in coding.

**4.1 General Technology Choices**

* **Programming Languages:**
  + **Python:** 3.11.8 (FastAPI backend, data processing, AI integrations). Type hints (PEP 484) are used, generated/assisted by AI.
  + **JavaScript/TypeScript:** Node.js v20.12.2 (LTS) and TypeScript 5.4.5 (Next.js frontend).
* **Backend Frameworks:**
  + **FastAPI:** 0.110.1. AI tools are heavily utilized for generating API endpoints, data models, and business logic.
  + **Other Python Libraries:** Pydantic 2.7.1, Uvicorn 0.29.0, Pandas 2.2.2, Requests 2.31.0. All integrated and debugged with AI assistance.
* **Frontend Frameworks/Libraries:**
  + **Next.js:** 14.2.3. AI tools assist in component generation, routing setup, and data fetching logic.
  + **React:** 18.2.0.
  + **Other React-specific Libraries:** React Query 5.31.0, Zustand 4.5.2, Tailwind CSS 3.4.3, Shadcn/UI 0.10.0, Chart.js 4.4.2. AI tools generate configurations and integrate these libraries into components.
* **Database:**
  + **PostgreSQL:** Supabase Cloud 15.x. SQL schema definitions and migration scripts are primarily generated by AI based on PRD specifications.
  + **ORM/Query Builder:** Supabase Python client 2.1.0. AI assists in crafting ORM calls and raw SQL queries.
* **Authentication:**
  + **Supabase Auth:** Managed service. AI tools guide implementation of login/logout flows and JWT handling.
  + **Authentication Methods (MVP):** Email/Password.

**4.2 External APIs & Services**

* **Shopify:** API Version 2024-07. Direct HTTP requests using Python requests (AI generates request structures).
* **ZenRows:** Endpoint https://api.zenrows.com/v1/get. Direct HTTP requests (AI generates scraping logic).
* **Google Trends (pytrends):** pytrends 4.9.2. AI assists in pytrends API calls and data parsing.
* **API Deck:** Endpoint https://api.apideck.com/v1/accounting. Direct HTTP requests (AI assists in mapping data).
* **Generative Media Stack:**
  + **Azure OpenAI:** gpt-4-turbo-2024-04-09. AI tools are directly calling this API for script generation.
  + **VEED.io:** API Endpoint https://api.veed.io/v1/video/generate. AI assists in structuring API calls and handling video URLs.
  + **11Eleven Labs:** API Endpoint https://api.elevenlabs.io/v1/text-to-speech/{voice\_id}. AI helps with voice synthesis API calls.
  + **Fal.ai / Photoroom:** Not directly integrated for MVP, but AI will assist if simple visual overlays are attempted within the limited time.

**4.3 Hosting & Deployment Environment (Azure)**

* **Overall Architecture Diagram (Text-based):**
* ┌───────────────────────────┐ ┌───────────────────────────┐
* │ Azure Static Web Apps │ │ Azure Blob Storage │
* │ (Next.js Frontend) │◄───────►│ (Generated Videos, │
* └───────────────────────────┘ │ CSV Uploads) │
* ▲ └───────────────────────────┘
* │ (API Calls) ▲
* ▼ │ (Data for AI)
* ┌───────────────────────────┐ ▼
* │ Azure App Service │ ┌───────────────────────────┐
* │ (FastAPI Backend) │◄───────►│ Supabase Cloud │
* └───────────────────────────┘ │ (PostgreSQL Database, │
* ▲ │ Auth, Storage) │
* │ (Trigger/Queue) └───────────────────────────┘
* ▼
* ┌───────────────────────────┐
* │ Azure Functions │
* │ (Python Runtime) │
* │ - Daily Cron Jobs │
* │ - Background Tasks │
* │ (Scraping, Trends, AI) │
* └───────────────────────────┘
* **Specific Azure Services:**
  + **Frontend:** Azure Static Web Apps.
  + **Backend:** Azure App Service (Linux Container, B1 Basic SKU). AI tools assist in generating Dockerfiles and deployment configurations.
  + **Data Processing/Background Jobs:** Azure Functions (Python Runtime, Consumption Plan). AI assists in generating serverless function code and trigger definitions.
  + **Database:** Supabase Cloud Managed Service (EU West).
  + **Secret Management:** Azure Key Vault. AI will help manage secret retrieval in code.
  + **Storage:** Azure Blob Storage. AI assists in generating file upload/download logic.
  + **Logging & Monitoring:** Azure Monitor, Application Insights, Log Analytics Workspace. AI helps interpret logs and set up alerts.

**4.4 Development & Operations (DevOps)**

* **Version Control System:** GitHub. Branching strategy (main, dev, feature/) is adhered to.
* **CI/CD Pipeline:** GitHub Actions. AI tools are used to generate workflow files for CI/CD steps (linting, testing, building Docker images, deploying to Azure).
* **Local Development Environment:** Docker Desktop, Node.js v20.12.2, Python 3.11.8, npm, pip. AI tools provide setup instructions and environment troubleshooting.
* **Error Reporting & Alerting:** Application Insights, Azure Monitor Alerts (email). AI helps analyze logs for error patterns and suggest fixes.
* **Security Practices:** Dependency scanning (AI-assisted npm audit/pip-audit), basic security headers (AI-generated fastapi-security middleware), FastAPI rate limits (fastapi-limiter). AI helps identify and mitigate common vulnerabilities.

**Section 5: Timeline and Deadlines**

This section defines the precise schedule, key milestones, and deadlines for the hackathon project, providing a roadmap for efficient development and delivery within the tight timeframe, with the understanding that all coding will be AI-assisted.

**5.1 Overall Hackathon Duration:**

* **Exact Start Date & Time:** Friday, May 31, 2025, 09:00 AM BST
* **Exact End Date & Time:** Saturday, June 1, 2025, 05:00 PM BST
* **Total Duration:** 32 hours

**5.2 Key Milestones & Deliverables:**

This schedule is highly ambitious and leverages the rapid code generation capabilities of AI tools. The single developer's primary role is prompt engineering, code review, integration, and debugging.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Milestone #** | **Milestone Title** | **Start Time (from 09:00 AM Fri)** | **End Time (from 09:00 AM Fri)** | **Duration (Hours)** | **Responsible (AI Assisted)** | **Dependencies** | **Success Criteria** |
| **0** | **Project Setup & Planning** | 09:00 AM (Fri) | 10:00 AM (Fri) | 1 | Solo Developer (AI Assisted) | None | All development environments set up, codebase initialized, initial Supabase project created, clear task breakdown. AI tools assist in generating boilerplate code and setup scripts. |
| **1** | **Core Shopify Integration** | 10:00 AM (Fri) | 02:00 PM (Fri) | 4 | Solo Developer (AI Assisted) | None | POST /api/shopify-sync endpoint (AI-generated) successfully connects to a Shopify dev store, ingests min 50 active products and 500 sales records into Supabase products and sales tables. Data integrity verified via Supabase console. |
| **2** | **Competitor Pricing Ingestion** | 02:00 PM (Fri) | 06:00 PM (Fri) | 4 | Solo Developer (AI Assisted) | Milestone 1 | scrape\_competitors.py (AI-generated) scrapes min 5 competitor URLs via ZenRows. POST /api/competitor-prices (AI-generated) ingests data, populating competitor\_prices table for min 10 SKUs. |
| **3** | **Google Trends & Mock Social** | 02:00 PM (Fri) | 06:00 PM (Fri) | 4 | Solo Developer (AI Assisted) | Milestone 1 | fetch\_trends.py (AI-generated) fetches Google Trends for min 5 keywords and generates mock social scores. POST /api/trends (AI-generated) ingests data, populating trend\_insights table for min 10 SKUs. |
| **4** | **Pricing & Trend Engine Logic** | 06:00 PM (Fri) | 10:00 PM (Fri) | 4 | Solo Developer (AI Assisted) | Milestones 1, 2, 3 | Backend logic (AI-generated) calculates recommended\_price, pricing\_reason, final\_score, and label for all SKUs. recommended\_prices table populated. Manual verification on 10 samples. |

**5.2 Key Milestones & Deliverables:**

This schedule is aggressive and assumes focused work leveraging AI coding tools. The solo developer's primary role is prompt engineering, integration, and debugging.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Milestone #** | **Milestone Title** | **Start Time (from 09:00 AM Fri)** | **End Time (from 09:00 AM Fri)** | **Duration (Hours)** | **Responsible (AI Assisted)** | **Dependencies** | **Success Criteria** |
| **0** | **Project Setup & Planning** | 09:00 AM (Fri) | 10:00 AM (Fri) | 1 | Solo Developer (AI Assisted) | None | Dev environments set up, codebase initialized, initial Supabase project created, clear task breakdown. AI tools assist in generating boilerplate code and setup scripts. |
| **1** | **Core Shopify Integration** | 10:00 AM (Fri) | 02:00 PM (Fri) | 4 | Solo Developer (AI Assisted) | None | POST /api/shopify-sync endpoint (AI-generated) successfully connects to a Shopify dev store, ingests min 50 active products and 500 sales records into Supabase products and sales tables. Data integrity verified via Supabase console. |
| **2** | **Competitor Pricing Ingestion** | 02:00 PM (Fri) | 06:00 PM (Fri) | 4 | Solo Developer (AI Assisted) | Milestone 1 | scrape\_competitors.py (AI-generated) scrapes min 5 competitor URLs via ZenRows. POST /api/competitor-prices (AI-generated) ingests data, populating competitor\_prices table for min 10 SKUs. |
| **3** | **Google Trends & Mock Social** | 02:00 PM (Fri) | 06:00 PM (Fri) | 4 | Solo Developer (AI Assisted) | Milestone 1 | fetch\_trends.py (AI-generated) fetches Google Trends for min 5 keywords and generates mock social scores. POST /api/trends (AI-generated) ingests data, populating trend\_insights table for min 10 SKUs. |
| **4** | **Pricing & Trend Engine Logic** | 06:00 PM (Fri) | 10:00 PM (Fri) | 4 | Solo Developer (AI Assisted) | Milestones 1, 2, 3 | Backend logic (AI-generated) calculates recommended\_price, pricing\_reason, final\_score, and label for all SKUs. recommended\_prices table populated. Manual verification on 10 samples. |
| **5** | **Basic Frontend Dashboard** | 10:00 PM (Fri) | 02:00 AM (Sat) | 4 | Solo Developer (AI Assisted) | Milestone 4 | Next.js dashboard renders a table of products displaying: SKU, Title, Current Price, Inventory, Competitor Range, Recommended Price, and Trend Label. Basic filtering/sorting. User login via Supabase Auth. All UI code AI-generated. |
| **6** | **Generative Media Video Summary** | 10:00 AM (Sat) | 02:00 PM (Sat) | 4 | Solo Developer (AI Assisted) | Milestone 4, VEED.io/11Labs API keys | Backend (AI-generated) identifies top 3 insights. Azure OpenAI generates script. VEED.io and 11Eleven Labs APIs successfully called to generate a 45-60 sec video. Video URL stored. Frontend displays embedded video player and transcript. |
| **7** | **Polish UI/UX & Demo Preparation** | 02:00 PM (Sat) | 04:00 PM (Sat) | 2 | Solo Developer (AI Assisted) | Milestones 5, 6 | Dashboard UI refined (AI-generated styling/component tweaks). Error messages user-friendly. Video player controls active. Demo script prepared and presentation rehearsed. |
| **8** | **Final Demo Rehearsal & Submission** | 04:00 PM (Sat) | 05:00 PM (Sat) | 1 | Solo Developer (AI Assisted) | Milestone 7 | Full end-to-end demo walkthrough completed without critical issues. Application deployed and accessible. Submission materials (codebase, presentation) ready. |

**5.3 Daily Schedule:**

Given the 32-hour timeframe and reliance on AI, this schedule is extremely condensed. The solo developer's time is primarily spent on prompt engineering, integrating generated code, and debugging.

* **Friday, May 31, 2025 (Day 1: 09:00 AM - 12:00 AM = 15 hours work)**
  + **09:00 AM - 10:00 AM:** **Kick-off & Setup (M0)**
    - Solo Developer: Define high-level prompts for initial project structure, boilerplate, and environment setup scripts. Review AI-generated code.
  + **10:00 AM - 02:00 PM:** **Shopify Integration & Core Data Sync (M1)**
    - Solo Developer: Prompt AI for Shopify OAuth implementation, API client code, data ingestion logic, and Supabase schema definitions. Integrate and debug generated code.
  + **02:00 PM - 06:00 PM:** **External Data Ingestion (M2 & M3 - Parallel Prompts)**
    - Solo Developer: Prompt AI for ZenRows scraping script and data parsing. Simultaneously prompt for pytrends integration and mock social score generation. Integrate and debug.
  + **06:00 PM - 10:00 PM:** **Core Logic Engines (M4)**
    - Solo Developer: Prompt AI for pricing optimization algorithms and trend classification logic based on detailed rules. Integrate calculated outputs into database.
  + **10:00 PM - 12:00 AM (Midnight):** **Initial Dashboard UI (M5 - Start)**
    - Solo Developer: Prompt AI for Next.js project setup, basic dashboard components, data display tables, and Supabase Auth integration. Integrate and test UI.
* **Saturday, June 1, 2025 (Day 2: 12:00 AM - 05:00 PM = 17 hours, including rest)**
  + **12:00 AM - 02:00 AM:** **Continue Dashboard UI (M5 - Finish)**
    - Solo Developer: Continue prompting AI for UI refinements (filtering, sorting). Debug frontend-backend integration.
  + **02:00 AM - 09:00 AM:** ***Solo Developer Rest / Buffer Time (7 hours)***
    - Crucial for mental clarity and error reduction, especially when working with AI outputs.
  + **09:00 AM - 10:00 AM:** **Morning Sync & Refocus**
    - Solo Developer: Review overnight progress, identify critical path for remaining hours. Adjust AI prompting strategy.
  + **10:00 AM - 02:00 PM:** **Generative Media Integration (M6)**
    - Solo Developer: Prompt AI for insight aggregation logic, Azure OpenAI script generation, VEED.io and 11Eleven Labs API calls. Integrate video player in frontend. Debug API calls.
  + **02:00 PM - 04:00 PM:** **UI/UX Polish & Error Handling (M7)**
    - Solo Developer: Prompt AI for styling improvements (Tailwind/Shadcn), user feedback mechanisms (loading states, error messages). Final testing of core flows.
  + **04:00 PM - 05:00 PM:** **Final Demo Preparation & Submission (M8)**
    - Solo Developer: Final end-to-end tests. Ensure deployment is stable. Prepare presentation notes focusing on AI-assisted development.

**5.4 Contingency Planning:**

The reliance on AI introduces unique risks and opportunities.

* **Risk:** AI Hallucinations / Incorrect Code Generation
  + **Contingency:** Aggressive code review of *all* AI-generated code. Immediate unit/integration testing where possible. If AI provides persistently incorrect code for a complex problem, simplify the feature or revert to a known working state. Prioritize smaller, well-defined prompts.
* **Risk:** AI Tool Outages / Rate Limits
  + **Contingency:** Have multiple AI coding tools (e.g., primary LLM, secondary LLM, specialized code generation tools) as backup. Rely on locally available documentation for core frameworks.
* **Risk:** API Outages (External APIs)
  + **Contingency:** Implement robust retry mechanisms. Use mocked data (as planned for social mentions) for critical paths if an API is persistently down. Prioritize core functionality.
* **Risk:** Unexpected Bugs / Integration Challenges (despite AI assistance)
  + **Contingency:** Utilize AI for debugging (explaining error messages, suggesting fixes). If a feature proves too complex or buggy, prune its scope and focus on demonstrability of core value.
* **Risk:** Time Overrun (due to AI prompting/debugging overhead)
  + **Contingency:** Ruthlessly de-scope any non-essential features. Focus strictly on the defined MVP. Utilize the dedicated rest/buffer time strategically for critical tasks.
* **Buffer Time:** The schedule includes a dedicated 7-hour rest period on Day 2, which also serves as a crucial buffer. The final 3 hours are for polish and demo prep, which can be compressed if development tasks spill over.

**Section 6: Success Criteria**

This section outlines the specific, measurable criteria that will determine the project's success, encompassing functional completeness, technical robustness, user experience, and readiness for demonstration, while acknowledging the unique AI-driven development approach.

**6.1 Core Functional Success Criteria (Minimum Viable Product - MVP)**

* **Shopify Integration:**
  + A Shopify development store successfully connects via OAuth, with its access\_token securely stored.
  + The POST /api/shopify-sync endpoint successfully ingests **at least 50 active product SKUs** (details: title, variant, prices, inventory, cost) and **500 historical sales records** from a Shopify store into the Supabase database within 10 minutes.
  + Supabase tables (stores, products, sales) are accurately populated and reflect synced data.
* **Competitor Pricing Data:**
  + The scrape\_competitors.py script successfully scrapes prices for **at least 5 specified competitor URLs** (from config/competitor\_urls.json) using ZenRows.
  + min\_price and max\_price are correctly derived and stored in the competitor\_prices table for at least **10 selected SKUs**.
  + Scraped prices are within +/- 5% of actual competitor site values.
* **Market Trend Data:**
  + The fetch\_trends.py script successfully fetches Google Trends data for **at least 5 product keywords**.
  + A mock social\_score is generated for all relevant products.
  + google\_trend\_index, social\_score, final\_score, and label are calculated and stored in trend\_insights for at least **10 selected SKUs**.
* **Pricing Engine:**
  + Backend logic correctly calculates a recommended\_price and pricing\_reason for at least **10 sample SKUs** based on defined rules (cost-plus, competitor, trend).
  + Recommendations are stored in the recommended\_prices table.
* **Trend Engine:**
  + Backend logic correctly assigns a label (Hot, Rising, Steady, Declining) to at least **10 sample SKUs** based on final\_trend\_score.
* **Generative Media Insights:**
  + Backend identifies the **top 3 most impactful insights** (e.g., underpriced, overstocked, hot product).
  + Azure OpenAI generates a coherent, relevant 45-60 second video script for these insights.
  + VEED.io and 11Eleven Labs APIs are successfully used to generate an MP4 video with voiceover, and the video URL is stored.
* **Frontend Dashboard:**
  + User (logged in via Supabase Auth) can view a functional dashboard displaying a list of products with current\_price, inventory\_level, competitor\_price range, recommended\_price, and trend\_label.
  + Basic filter/search for products by SKU/title is functional.
  + The latest generated AI Advisor video is playable directly within the dashboard.

**6.2 Technical Success Criteria**

* **Deployment:** The entire application (Frontend, Backend, Background Jobs, Database) is successfully deployed and running on Azure services (Static Web Apps, App Service, Functions) and Supabase Cloud, accessible via a public URL.
* **Code Quality (for Hackathon):**
  + The codebase, largely generated by AI, is well-organized, understandable, and passes basic linting checks (aided by AI).
  + No critical bugs or crashes occur during core demonstrations.
  + The CI/CD pipeline (GitHub Actions) successfully triggers and completes builds/deployments.
* **Performance (for Hackathon):**
  + Core API endpoints respond within reasonable times (<500ms for dashboard loads, <10s for data sync triggers).
  + Video generation (end-to-end, script to URL) completes within a demonstrable timeframe (e.g., under 5 minutes).
* **Security:** API keys and sensitive credentials are stored securely (Azure Key Vault, GitHub Secrets). Basic API rate limits are active.

**6.3 User Experience & Presentation Success Criteria**

* **Ease of Onboarding:** A new user (hackathon judge) can connect a Shopify store and see initial data populated on the dashboard within 5 minutes of starting the demo.
* **Clarity of Insights:** Pricing recommendations and trend insights are clearly presented and immediately understandable on the dashboard.
* **Advisor Video Impact:** The AI Advisor video effectively summarizes key insights in a compelling, professional manner. Audio is clear, and visuals are relevant.
* **Demo Readiness:** The application is stable and robust for a live demonstration. The presentation is clear, concise, compelling, and effectively highlights problem-solution fit and the unique AI-driven development approach.

**6.4 Hackathon-Specific & "Bonus" Success Criteria**

* **Completion of Defined MVP Scope:** All functional criteria in 6.1 are demonstrably met.
* **Innovative Use of AI/APIs:** Successfully integrating Azure OpenAI, VEED.io, 11Eleven Labs, ZenRows, and API Deck to create a novel and impactful retail solution, specifically highlighting the AI-assisted coding process.
* **Scalability Potential:** The chosen architecture (serverless functions, managed databases, microservices approach for different data sources) demonstrates clear pathways for future scalability.
* **"Wow" Factor:** The generative video feature genuinely impresses judges and clearly conveys the product's unique value proposition.
* **"Painfully Detailed" PRD:** This document itself demonstrates thorough planning and communication, reflective of a well-managed project, even with AI as the primary coder.

**Section 7: Assumptions and Constraints**

This section explicitly lists the assumptions made during planning and development, along with the constraints and limitations that define the project's scope for the hackathon MVP, critically considering the AI-driven development process.

**7.1 Assumptions**

* **AI Tool Performance & Reliability:**
  + It is assumed that the various AI coding tools (e.g., LLMs for code generation, debugging aids) will perform adequately in generating functional, correct, and integratable code for all defined features.
  + It is assumed that the solo developer's ability to prompt, review, integrate, and debug AI-generated code will be efficient enough to meet the tight timeline.
  + It is assumed that AI tools can handle complex integrations with external APIs and resolve unforeseen issues that arise during this process.
* **Data Availability & Quality:**
  + Shopify stores will have sufficient active product and sales data.
  + variant.sku is consistently populated across Shopify, CSV, and accounting data for reliable matching.
  + variant.cost from Shopify or API Deck is reasonably accurate.
  + Competitor product pages have stable HTML structures for ZenRows scraping.
  + Product titles are suitable keywords for Google Trends.
* **External API Stability & Rate Limits:**
  + All external APIs (Shopify, ZenRows, Azure OpenAI, VEED.io, 11Eleven Labs, API Deck) will remain operational and within documented rate limits throughout the hackathon.
  + API response times will be acceptable for the planned syncs and video generation.
  + pytrends will reliably fetch data without blocking.
* **Authentication & Permissions:**
  + Shopify OAuth scopes (read\_products, read\_inventory, read\_orders) are sufficient.
  + API Deck provides essential item data for cost price.
* **Resources:**
  + Necessary API keys and developer accounts are pre-configured and accessible.
  + Consistent internet access and adequate hardware are available.
* **MVP Scope Definition:**
  + The defined MVP scope is achievable within 32 hours *with AI assistance*.
  + Mocked social trend data is acceptable for MVP.
  + CSV upload is a fallback, not primary.
* **Pricing & Trend Logic:**
  + Simple rule-based pricing and trend logic is sufficient for MVP.
  + All prices are in GBP; no currency conversion required.

**7.2 Constraints**

* **Time Constraint:** Absolute maximum development time is **32 hours**. This is the primary constraint shaping scope and complexity.
* **Resource Constraints:**
  + **Single Developer:** All development, integration, and debugging is performed by one individual, albeit augmented by AI tools. This limits parallel work streams to what can be managed through prompt engineering and concurrent AI responses.
  + Utilization of **free/trial tiers** of services where possible.
  + No dedicated DevOps/SRE; all infrastructure and deployment tasks are handled by the solo developer, leveraging AI for script generation.
* **Scope Constraint (Explicitly Out of Scope for MVP):**
  + **Real-time data synchronization:** Batch-based syncs only.
  + **Complex accounting integration:** Only cost\_price extraction.
  + **User-configurable pricing rules:** Rules are hardcoded.
  + **Advanced analytics/reporting:** Basic dashboard, no complex charts/drill-downs.
  + **Historical competitor/trend data:** Only latest data stored.
  + **Refund/Discount reconciliation** for sales.
  + **Multi-store management UI** (focus on one store per user).
  + **Advanced AI features:** No fine-tuning, no custom visual overlays beyond VEED.io's templates.
  + **User feedback/communication features.**
  + **Comprehensive error handling dashboard/alerts beyond basic logging.**
  + **Public-facing API.**
  + **Full cross-browser/device compatibility** (focus on modern desktop).

**7.3 Dependencies**

* **External API Access:** Active accounts and valid API keys/credentials for all external services (Shopify Partner, ZenRows, Azure OpenAI, VEED.io, 11Eleven Labs, API Deck) must be provisioned.
* **Supabase Project:** Must be set up with initial schema.
* **Azure Infrastructure:** Basic Azure resources (Resource Group, App Service, Static Web Apps, Functions) provisioned or easily deployable via AI-generated scripts.

**Section 8: Future Considerations / Roadmap**

This section outlines the potential evolution of the product beyond the hackathon MVP, detailing planned enhancements, new features, and strategic growth areas, with a particular emphasis on the continued, evolving role of AI in future development.

**8.1 Phase 2: Post-Hackathon Development (Short-Term Enhancements)**

Following the hackathon, the focus will be on hardening the MVP and expanding its core value proposition. AI tools will be crucial in rapidly implementing these enhancements.

* **Expanded Data Integrations:**
  + **Accounting Software:** Prioritize **Sage and additional Xero modules** (beyond just items) via API Deck. AI tools will generate specific API client code and data mapping logic.
  + **Other E-commerce Platforms:** Integrate **WooCommerce (REST API)** and **BigCommerce**. AI will assist in developing robust, platform-specific data connectors and ensuring data consistency.
  + **Third-Party Data Sources:** Integrate **Google Analytics 4 (GA4)** for traffic and conversion trends. AI will generate data pipelines to extract and integrate GA4 metrics into trend insights. Explore **Ad Platform APIs** (e.g., Google Ads, Facebook Ads) for spend and ROAS data.
* **Enhanced Pricing Engine:**
  + **Additional Pricing Strategies:** Implement **dynamic pricing** based on demand signals (e.g., real-time traffic from GA4, competitor price changes). AI could assist in developing rule engines and predictive models.
  + **Machine Learning Models:** Introduce basic **demand forecasting models** (scikit-learn or similar) to optimize inventory levels and suggest future pricing. AI will guide model selection, feature engineering, and training.
  + **User-Configurable Rules:** Develop a UI for retailers to define pricing rules (e.g., min/max margin, competitor weighting, trend sensitivity) via natural language prompts, which AI will translate into backend logic.
* **Deeper Trend Analysis:**
  + **Sieve.tool Integration:** Full integration with Sieve.tool (or similar) for actual social media mentions, sentiment analysis, and competitor brand monitoring. AI will interpret and synthesize this rich dataset.
  + **External Event Correlation:** Integrate public data sources for holidays, major news events, or economic indicators, leveraging AI to identify correlations with product performance.
* **User Experience & Dashboard Improvements:**
  + **Interactive Dashboards:** Introduce granular sales charts, inventory aging reports, and interactive competitor price history graphs. AI will generate Chart.js configurations and D3.js components.
  + **Guided Onboarding:** Improve onboarding with AI-driven interactive tutorials and personalized setup recommendations.
  + **Customizable Alert System:** Allow users to define custom alerts (e.g., low stock, sudden price drops, competitor changes) with AI-generated notification logic.
  + **Enhanced Mobile Responsiveness:** Comprehensive optimization for mobile and tablet devices.
* **Generative Media Enhancements:**
  + **Customization:** Enable users to select voice options, background music, visual styles, and branding for videos. AI will translate these preferences into API parameters.
  + **Targeted Videos:** Generate videos for specific product categories, collections, or individual items based on user selection.
  + **Multi-language Support:** Offer multi-language voiceovers and script generation via AI translation capabilities.
  + **Other Generative Media:** Generate marketing copy (e.g., product descriptions, social media posts, email snippets) based on performance insights.

**8.2 Phase 3: Long-Term Vision (Strategic Growth)**

The long-term vision is to establish the Retail AI Advisor as the leading AI-powered intelligence platform for SMEs, transforming reactive businesses into proactive, data-driven entities.

* **Market Expansion:**
  + **Vertical Expansion:** Adapt the platform for niche retail verticals (e.g., digital products, services, bespoke manufacturing), understanding their unique data and insight needs through AI-driven market analysis.
  + **Internationalization:** Full multi-currency, multi-language UI, and integration with region-specific data sources (local competitors, regional trends). AI will manage the complexity of global data aggregation and localization.
* **AI-Driven Automation:**
  + **Automated Price Execution:** Implement optional, user-approved automatic pushing of recommended price changes back to e-commerce platforms. AI will manage safeguards and audit trails.
  + **Inventory Reordering Automation:** Fully automate purchase order suggestions based on AI-driven demand forecasts, supplier lead times, and optimal stock levels.
  + **Automated Ad Campaign Suggestions:** Integrate directly with ad platforms to provide AI-driven recommendations for ad spend allocation, targeting, and creative optimization based on product performance.
* **Platform & Ecosystem:**
  + **App Store Integration:** Become a premier app on major e-commerce app stores (Shopify, BigCommerce), enabling seamless installation and greater reach.
  + **Open API & Developer Platform:** Expose a robust, well-documented API for third-party developers to build custom applications and integrations on top of the generated insights. AI could assist in API design and SDK generation.
  + **Retailer Community:** Build a collaborative platform where retailers can share anonymized insights, best practices, and learn from each other.
* **Monetization Strategy:**
  + **Tiered Subscriptions:** Implement subscription tiers based on SKU count, monthly sales volume, connected integrations, or access to advanced features (e.g., ML forecasting, real-time alerts).
  + **Premium Add-ons:** Offer premium features like dedicated account management, custom reporting, and advanced analytics consultations.

**8.3 AI-Driven Development: Future Outlook**

The hackathon is just the beginning of a profound shift in software development enabled by AI. The reliance on AI tools for coding will not diminish but evolve into a more sophisticated partnership.

* **Evolution of AI Reliance:** The solo developer's role will shift from managing AI for code generation to becoming an "AI Orchestrator" or "AI Architect." This involves designing higher-level system architectures, defining complex problem statements for AI, managing the integration of increasingly autonomous AI agents, and focusing on quality assurance and strategic oversight.
* **New AI-Driven Development Tools:**
  + **Autonomous Agent Development:** Explore leveraging multi-agent AI systems where different AI agents collaborate on specific tasks (e.g., one AI for frontend, one for backend, one for testing), reporting to the solo developer.
  + **AI-Driven Testing & Validation:** AI tools will not only generate code but also automatically create comprehensive unit, integration, and end-to-end test suites, significantly reducing manual QA effort and increasing code reliability.
  + **AI for Project Management:** AI could assist in dynamic timeline adjustments, proactive identification of blockers, and intelligent task re-prioritization, optimizing the development workflow.
  + **Self-Healing Code:** AI could be trained to identify and suggest fixes for production bugs autonomously, potentially leading to "self-healing" application components.
  + **AI for Architecture & Design:** AI could move into assisting with high-level architectural decisions, suggesting optimal database designs, microservice boundaries, and deployment strategies.
* **Long-Term Implications for the "Single Developer" Model:** The ambition is to prove that a single highly skilled developer, leveraging state-of-the-art AI tools, can build, maintain, and scale a complex SaaS product with the efficiency and quality typically associated with a small, specialized team. This model maximizes individual productivity and allows for rapid iteration and adaptation to market changes, positioning the solo developer as a product visionary and strategic implementer.

This completes the comprehensive Product Requirements Document for your Retail AI Advisor Hackathon MVP. Good luck with the hackathon!